05

¿Qué captura?

Ya hemos visto en este curso dos formas de capturar varias excepciones a través del bloque *catch*. (1) La forma tradicional, que ha funcionado desde el comienzo de Java, simplemente repite el bloque *catch* para cada excepción:

```
try {
    metodoPeligrosoQuePuedeLanzarVariasExcepciones();
} catch(MiExcepcion ex) {
    ex.printStackTrace();
} catch(OtraExcepcion ex) {
    ex.printStackTrace();
}
```

COPIA EL CÓDIGO

Y (2) la forma más actual, que se introdujo en Java 7, le permite definir las diversas excepciones en el mismo bloque (*multi catch*):

```
try {
    metodoPeligrosoQuePuedeLanzarVariasExcepciones ();
} catch(MiExcepcion | OtraExcepcion ex) {
    // multi-catch
    ex.printStackTrace();
}
```

COPIA EL CÓDIGO

Encontrarás ambas formas en tu día a día como desarrollador Java. Ahora, vea la firma del "método peligroso" en cuestión:

```
//funciona, podemos colocar dos excepciones en la firma

public void metodoPeligrosoQuePuedeLanzarVariasExcepcion
    // código omitido
}
```

COPIA EL CÓDIGO

Vimos otra variación más de *catch*, no sintáctica, sino conceptual. ¿Qué declaración a continuación se puede utilizar para capturar todas las excepciones de este "método peligroso"?

Importante: ¡asumiendo que ambas excepciones son de tipo *checked*!

A

```
try {
    metodoPeligrosoQuePuedeLanzarVariasExcepciones
}
```

```
try {
    metodoPeligrosoQuePuedeLanzarVariasExcepciones
} catch(Exception ex) {
    ex.printStackTrace();
}
```

Correcto. Creamos un *catch* genérica que captura cualquier excepción, incluidas las excepciones *checked*.
Esto puede parecer una buena práctica, pero generalmente no lo es. Como regla general, siempre trate de ser lo más específico posible en el bloque *catch* favoreciendo multiples bloques de *catch* o utilizando *multi-catch*.

C

```
try {
    metodoPeligrosoQuePuedeLanzarVariasExcepciones
} catch(RuntimeException ex) {
    ex.printStackTrace();
}
```

PRÓXIMA ACTIVIDAD